Technical Contribution 2: Systems & Security Engineer – Contributor B

Overview

This document explains the technical contributions made by Contributor B, who focused on building critical security systems for AI Shield. The contributions include quarantine systems, secure deletion, permission restrictions, background scanning, OS-level operations, Snort IDS integration, network monitoring, URL blocking, and threat action logic.

---

1. Quarantine System

What is a Quarantine System?

A quarantine system is like a secure isolation room for suspicious files. When a file is detected as potentially dangerous, instead of deleting it immediately, the system moves it to a special protected location where it cannot harm the system but can still be recovered if needed.

How It Works (Simple Explanation)

- **Detection**: When AI Shield finds a suspicious file, it marks it as a threat
- **Isolation**: The file is moved to a special quarantine folder (`~/.quarantine`)
- **Protection**: The file is renamed with a special format and locked with restricted permissions
- **Tracking**: All information about the file is saved in a database for future reference
- **Recovery**: If the file turns out to be safe, it can be restored to its original location

Key Features

- **Obfuscated Filenames**: Files are renamed using a hash and timestamp (e.g., `Qa3f2b1c_1704067200.quarantine`) to prevent accidental execution

- **Permission Locking**: Files are locked using OS-level permissions:

- **Windows**: Uses `icacls` to deny read/write/execute permissions
- **Linux/macOS**: Uses `chmod` to remove all permissions (000) or set read-only (444)

- **Metadata Storage**: SQLite database tracks:

- Original file path and name
- SHA256 hash for verification
- Timestamp and user who quarantined it
- Reason for quarantine
- Threat severity and source

- **Restore Capability**: Files can be restored to original location or a new path

- **Permanent Deletion**: Option to permanently delete quarantined files after review

Technical Implementation Points

- Uses `quarantine_manager.py` module for core functionality
- Cross-platform support (Windows, Linux, macOS)
- Atomic file operations (move or copy+delete)
- Automatic permission management
- JSON metadata files for recovery information
- Database integration with threat records

Questions & Answers

Q: Why quarantine instead of deleting files immediately?

A: Quarantine allows administrators to review suspicious files before permanent deletion. Sometimes files are false positives, and quarantine provides a safety net to recover them.

Q: How secure is the quarantine folder?

A: Very secure. Files are:

- Moved to a hidden directory (`~/.quarantine`)
- Renamed with obfuscated names
- Locked with restrictive permissions (no execute, no write)
- Tracked in a database with metadata

Q: Can quarantined files still run?

A: No. Files are:

- Renamed with `.quarantine` extension (prevents execution)
- Locked with no execute permissions
- Isolated in a protected directory

Q: What happens if I need to restore a file?

A: The system can restore files by:

- Looking up the original path from the database

- Unlocking file permissions

- Moving the file back to its original location

- Restoring the original filename

Q: How much space does quarantine use?

A: Quarantine uses the same amount of space as the original files. Files are moved, not copied, so there's no duplication. You can purge old quarantined files to free space.

---

2. Secure Delete System

What is Secure Delete?

Secure delete ensures that when files are removed, they are completely and safely deleted from the system. The system provides multiple methods to handle different scenarios, including moving files to Recycle Bin (for recovery) or permanent deletion.

How It Works (Simple Explanation)

- **Threat Detection**: System identifies a file as malicious or unwanted

- **Deletion Method Selection**: Chooses the best deletion method based on:

- User preference (Recycle Bin vs permanent)
- File permissions
- Operating system

- **Permission Handling**: Removes read-only attributes and access restrictions

- **Safe Deletion**: Uses multiple fallback methods to ensure deletion succeeds

- **Verification**: Confirms the file is actually deleted

Key Features

- **Multiple Deletion Methods**:

- **Recycle Bin/Trash** (preferred): Moves files to system trash for recovery
- **Permanent Deletion**: Completely removes files from filesystem
- **Elevated Deletion**: Uses admin privileges when needed

- **Cross-Platform Support**:

- **Windows**: Uses `send2trash`, `icacls`, `attrib`, PowerShell, CMD
- **Linux/macOS**: Uses `send2trash`, `chmod`, `sudo rm`

- **Permission Handling**: Automatically handles:

- Read-only files
- Files with restricted permissions
- Files requiring admin access

- **Fallback Mechanisms**: If one method fails, tries alternative methods

- **Audit Logging**: Records all deletion attempts and results

Technical Implementation Points

- Uses `delete_anomalies.py` script for deletion operations

- Supports `send2trash` library for cross-platform Recycle Bin support

- Multiple fallback methods for reliability

- Admin elevation support (Windows UAC, Linux sudo)

- Batch deletion support for multiple files

- Dry-run mode for testing without actual deletion

- Results logging to JSON files

Questions & Answers

Q: What's the difference between Recycle Bin and permanent deletion?

A:

- **Recycle Bin**: File is moved to a special folder where it can be recovered. Takes up disk space until emptied.

- **Permanent Deletion**: File is immediately removed from the filesystem and cannot be recovered easily.

Q: Can deleted files be recovered?

A: Files sent to Recycle Bin can be recovered until the bin is emptied. Permanently deleted files are harder to recover and may require special data recovery tools.

Q: What if a file can't be deleted?

A: The system tries multiple methods:

- Normal deletion
- Remove read-only attributes
- Grant full permissions
- Use admin/elevated privileges
- Use OS-specific commands (PowerShell, CMD, sudo)

Q: Is secure delete safe?

A: Yes. The system:

- Verifies file existence before deletion
- Handles permissions properly
- Uses safe deletion methods
- Logs all operations for audit

Q: Can I delete multiple files at once?

A: Yes. The system supports batch deletion of multiple threat files, with progress tracking and summary reports.

---

3. Permission Restriction System

What is Permission Restriction?

Permission restriction is a security feature that prevents suspicious files from being executed or modified by removing or limiting their access permissions at the operating system level.

How It Works (Simple Explanation)

- **Threat Detection**: File is identified as suspicious or malicious

- **Permission Analysis**: System checks current file permissions

- **Restriction Application**: Applies restrictive permissions based on security level:

- **Standard**: Read-only, no execute
- **Moderate**: Read-only with minimal restrictions
- **Strict**: Maximum restrictions (no permissions at all)

- **Verification**: Confirms permissions were applied successfully

Key Features

- **Three Restriction Levels**:

- **Standard**: Removes execute and write permissions (most common)
- **Moderate**: Read-only access only
- **Strict**: No permissions at all (maximum security)

- **Cross-Platform Support**:

- **Windows**: Uses `attrib` and `icacls` commands
- **Linux/macOS**: Uses `chmod` command

- **In-Place Protection**: Files can be restricted without moving them

- **Reversible**: Permissions can be restored if file is determined to be safe

- **Audit Trail**: Records permission changes with before/after states

Technical Implementation Points

- Uses `_set_restrictive_permissions()` function in `threat_actions.py`
- Windows: `attrib +R` for read-only, `icacls /deny` for execute/write
- Linux/macOS: `chmod 444` (read-only) or `chmod 000` (no permissions)
- Supports both file and directory restrictions
- Automatic permission restoration for restore operations

Questions & Answers

Q: Why restrict permissions instead of deleting?

A: Permission restriction provides immediate protection while allowing administrators to investigate the file. It's reversible and doesn't require moving files.

Q: Can restricted files still be read?

A: It depends on the restriction level:

- **Standard/Moderate**: Files can be read but not executed or modified
- **Strict**: Files cannot be read, written, or executed

Q: How do I restore file permissions?

A: The system can automatically restore permissions when:

- A file is determined to be safe
- A file is restored from quarantine
- An administrator manually requests restoration

Q: Does this work on all file types?

A: Yes, permission restriction works on all file types (executables, scripts, documents, etc.) across Windows, Linux, and macOS.

Q: What if permission restriction fails?

A: The system will:

- Log the error
- Try alternative methods
- Fall back to quarantine if restriction fails
- Report the issue to administrators

---

4. Background Scanner

What is a Background Scanner?

A background scanner continuously monitors the file system in real-time, automatically

scanning files as they are created, modified, or moved. It works silently in the background without interrupting the user.

How It Works (Simple Explanation)

- **File System Monitoring**: Watches specified directories for file changes

- **Event Detection**: Detects when files are:

- Created (new files)
- Modified (existing files changed)
- Moved (files relocated)

- **Automatic Scanning**: Scans detected files using ML anomaly detection

- **Threat Reporting**: Reports threats found during scanning

- **Continuous Operation**: Runs continuously in the background

Key Features

- **Real-Time Monitoring**: Uses `watchdog` library to monitor file system events

- **Intelligent Filtering**: Skips unnecessary files:

- System directories (node_modules, .git, __pycache__, venv)
- Temporary files
- Files larger than 100MB
- Already scanned files

- **Delayed Scanning**: Uses a delay mechanism to avoid scanning the same file multiple times

- **Continuous Scanning**: Periodically scans all files in monitored directories

- **Health Monitoring**: Automatic restart if the scanner stops working

- **Progress Tracking**: Reports scan progress and statistics

- **Threat Aggregation**: Collects threats and reports them periodically

Technical Implementation Points

- Uses `BackgroundScanner` class in `background.py`

- File system events handled by `BackgroundScanHandler`
- Asynchronous scanning for non-blocking operation
- Thread-based architecture for concurrent operations
- Health check thread for reliability
- Statistics tracking (files scanned, threats found, bytes scanned)
- Configurable scan delay and threat report intervals

Questions & Answers

Q: Does background scanning slow down my computer?

A: No. The scanner:

- Uses efficient file system monitoring
- Scans files asynchronously (non-blocking)

- Skips large files and system directories

- Uses intelligent caching to avoid re-scanning

Q: What directories are monitored?

A: You can configure which directories to monitor. By default, it excludes:

- System directories (node_modules, .git, __pycache__, venv)

- Temporary folders

- Very large files (>100MB)

Q: How often does it scan?

A: The scanner:

- Monitors files in real-time (immediate scanning on file changes)

- Performs full directory scans periodically (configurable interval)

- Reports threats at regular intervals (default: 1 minute)

Q: Can I pause or stop the scanner?

A: Yes. The scanner can be:

- Paused temporarily
- Stopped completely
- Restarted with new configuration
- Configured to monitor different directories

Q: What happens if the scanner crashes?

A: The system includes:

- Health check monitoring
- Automatic restart on failure
- Error logging and reporting
- Graceful recovery mechanisms

---

5. OS-Level Operations

What are OS-Level Operations?

OS-level operations are low-level system commands that interact directly with the operating system to perform security tasks that regular applications cannot do.

How It Works (Simple Explanation)

- **System Command Execution**: Runs operating system commands to perform security tasks
- **Permission Management**: Uses OS commands to modify file permissions
- **Process Management**: Monitors and controls system processes
- **Network Operations**: Manages network connections and firewall rules

- **System Integration**: Integrates with OS security features

Key Features

- **Cross-Platform Commands**:

- **Windows**: `attrib`, `icacls`, `takeown`, `netsh`, PowerShell
- **Linux/macOS**: `chmod`, `chattr`, `iptables`, `sudo`

- **File System Operations**:

- Permission changes
- File locking/unlocking
- Directory management

- **Process Operations**:

- Process monitoring
- Process termination
- Resource usage tracking

- **Network Operations**:

- Connection monitoring
- Firewall rule management
- DNS cache management

Technical Implementation Points

- Uses `subprocess` module for command execution
- Platform detection (`os.name` , `platform.system()`)
- Admin/elevated privilege detection
- Timeout handling for long-running operations
- Error handling and fallback mechanisms
- Cross-platform compatibility layer

Questions & Answers

Q: Why are OS-level operations needed?

A: Some security tasks require direct OS access:

- Changing file permissions

- Blocking network connections
- Managing system processes
- Integrating with OS security features

Q: Do OS-level operations require admin rights?

A: Some operations do, such as:

- Modifying system files
- Changing network settings
- Blocking URLs in hosts file
- Managing system processes

Q: Are OS-level operations safe?

A: Yes. The system:

- Validates all commands before execution

- Uses safe, well-known OS commands
- Includes error handling and timeouts
- Logs all operations for audit

Q: What if an OS operation fails?

A: The system:

- Tries alternative methods
- Falls back to safer operations
- Reports errors to administrators
- Continues with other operations

Q: Which operating systems are supported?

A: The system supports:

- Windows (Windows 10/11)

- Linux (various distributions)

- macOS (recent versions)

---

6. Snort IDS Integration

What is Snort IDS?

Snort is an open-source Intrusion Detection System (IDS) that monitors network traffic for suspicious activity and potential attacks. AI Shield integrates with Snort to receive and display security alerts.

How It Works (Simple Explanation)

- **Snort Monitoring**: Snort monitors network traffic in real-time

- **Alert Generation**: When suspicious activity is detected, Snort creates alerts
- **Alert Reading**: AI Shield reads alerts from Snort's log files
- **Alert Processing**: Alerts are parsed and formatted for display
- **Dashboard Integration**: Alerts are shown in the AI Shield dashboard

Key Features

- **Alert File Reading**: Reads from common Snort alert file locations:

- Linux: `/var/log/snort/alert`
- Windows: `C:/Snort/log/alert`
- Custom paths supported

- **Real-Time Updates**: Periodically reads latest alerts
- **Alert Formatting**: Parses and formats alerts for display
- **Integration**: Seamlessly integrates with AI Shield threat feed
- **Fallback Support**: Provides sample alerts if Snort is not configured

Technical Implementation Points

- Uses `snort.py` module for alert reading
- File-based alert reading (reads last 200 lines)
- Error handling for missing files
- Cross-platform path detection
- Integration with threat management system

Questions & Answers

Q: What is Snort used for?

A: Snort is an Intrusion Detection System that:

- Monitors network traffic

- Detects suspicious patterns
- Identifies potential attacks
- Generates security alerts

Q: Do I need to install Snort separately?

A: Yes. Snort is a separate tool that must be installed and configured. AI Shield reads alerts from Snort's log files.

Q: How does AI Shield integrate with Snort?

A: AI Shield:

- Reads alert files from Snort's default locations
- Parses and formats alerts
- Displays them in the dashboard
- Integrates alerts with the threat management system

Q: What if Snort is not installed?

A: If Snort is not installed or configured:

- AI Shield will show a message indicating Snort is not configured
- Other security features continue to work
- You can still use file scanning, URL blocking, and other features

Q: Can I use a custom Snort alert file location?

A: Yes. You can specify a custom path to the Snort alert file, and AI Shield will read from that location.

---

7. Network Monitoring

What is Network Monitoring?

Network monitoring tracks active network connections on the system, identifying which

processes are communicating with which remote addresses, and monitoring data transfer rates.

How It Works (Simple Explanation)

- **Connection Detection**: Monitors active network connections
- **Process Identification**: Identifies which processes are making connections
- **Remote Address Tracking**: Tracks remote IP addresses and ports
- **Traffic Analysis**: Monitors data transfer rates (bytes per second)
- **Alert Generation**: Alerts on suspicious connections

Key Features

- **Active Connection Tracking**: Monitors all active network connections
- **Process Information**: Shows which processes are making connections

- **Remote Address Monitoring**: Tracks remote IP addresses and ports
- **Traffic Metrics**: Monitors upload/download speeds
- **IP Blocking**: Can block suspicious IP addresses
- **Real-Time Updates**: Updates connection information in real-time

Technical Implementation Points

- Uses system network APIs for connection monitoring
- Process ID (PID) tracking for process identification
- Remote address and port extraction
- Traffic rate calculation
- Integration with threat management system

- WebSocket updates for real-time dashboard updates

Questions & Answers

Q: What information does network monitoring show?

A: Network monitoring shows:

- Active network connections
- Process names and IDs making connections
- Remote IP addresses and ports
- Data transfer rates (upload/download)
- Connection timestamps

Q: Can I block suspicious IP addresses?

A: Yes. The system allows you to:

- Block specific IP addresses

- Block entire IP ranges
- View blocked IP addresses
- Remove blocks if needed

Q: Does network monitoring slow down my network?

A: No. Network monitoring:

- Only observes connections (doesn't interfere)
- Uses efficient system APIs
- Minimal performance impact
- Runs in the background

Q: What are suspicious network connections?

A: Suspicious connections may include:

- Connections to known malicious IPs

- Unusual data transfer patterns

- Connections from unknown processes

- High-volume connections to unknown addresses

Q: Can I see historical network activity?

A: The system tracks recent network activity and displays it in the dashboard. Historical data is stored in the database for analysis.

---

8. URL Blocking

What is URL Blocking?

URL blocking prevents access to malicious or suspicious websites by modifying the system's hosts file or using other OS-level blocking mechanisms.

How It Works (Simple Explanation)

- **URL Analysis**: Analyzes URLs for risk factors (phishing, malware, suspicious patterns)
- **Risk Scoring**: Calculates a risk score for each URL
- **Blocking Decision**: Decides whether to block based on risk threshold
- **OS-Level Blocking**: Adds entries to the hosts file to redirect malicious domains
- **DNS Cache Flush**: Clears DNS cache to apply blocks immediately

Key Features

- **Automatic Blocking**: Automatically blocks URLs with high risk scores
- **Manual Blocking**: Allows manual blocking of specific URLs
- **Hosts File Modification**: Adds entries to system hosts file:

- Windows: `C:\Windows\System32\drivers\etc\hosts`
- Linux/macOS: `/etc/hosts`

- **DNS Cache Management**: Flushes DNS cache after blocking

- **Database Tracking**: Stores blocked URLs in database

- **Risk Assessment**: Comprehensive URL risk scoring

Technical Implementation Points

- Uses `webshield.py` module for URL blocking

- Hosts file parsing and modification

- Cross-platform hosts file path detection

- DNS cache flushing commands:

- Windows: `ipconfig /flushdns`
- macOS: `dscacheutil -flushcache`
- Linux: `systemd-resolve --flush-caches`

- Admin/elevated privilege handling

- Fallback methods for permission issues

Questions & Answers

Q: How does URL blocking work?

A: URL blocking:

- Analyzes URLs for risk factors

- Calculates a risk score

- If score exceeds threshold, blocks the URL

- Adds entry to hosts file (redirects to 127.0.0.1)

- Flushes DNS cache to apply immediately

Q: Can I unblock a URL?

A: Yes. You can:

- Remove entries from the blocked URLs list

- Manually edit the hosts file

- Use the dashboard to unblock URLs

Q: Does URL blocking require admin rights?

A: Yes. Modifying the hosts file requires administrator/elevated privileges on all operating systems.

Q: What URLs are automatically blocked?

A: URLs are automatically blocked if they:

- Have a high risk score (≥0.6 by default)

- Are categorized as phishing or malware

- Match known malicious patterns

Q: Can I block specific domains?

A: Yes. You can manually block:

- Specific URLs
- Entire domains
- Subdomains
- IP addresses (for direct IP access)

---

9. Threat Action Logic

What is Threat Action Logic?

Threat action logic is the decision-making system that determines what actions to take when threats are detected. It coordinates quarantine, deletion, permission restrictions, and other security responses.

How It Works (Simple Explanation)

- **Threat Detection**: System detects a threat (file, URL, network activity)
- **Threat Analysis**: Analyzes threat severity, source, and type
- **Action Selection**: Chooses appropriate action based on threat characteristics
- **Action Execution**: Executes the selected action (quarantine, delete, restrict, etc.)
- **Status Tracking**: Tracks action status and updates threat records

Key Features

- **Multiple Action Types**:

- **Quarantine**: Move file to secure location
- **Delete**: Remove file from system
- **Restrict**: Limit file permissions
- **Allow**: Mark file as safe (false positive)

- **Automated Decision Making**: Automatically selects actions based on threat severity

- **Manual Override**: Administrators can manually change actions
- **Bulk Operations**: Supports bulk actions on multiple threats
- **Action History**: Tracks all actions taken on threats
- **Database Integration**: Updates threat records with action status

Technical Implementation Points

- Uses `threat_actions.py` module for action handling
- Integration with quarantine system
- Integration with secure delete system
- Integration with permission restriction system
- Database updates for action tracking

- WebSocket notifications for real-time updates

- Error handling and rollback mechanisms

Questions & Answers

Q: How does the system decide what action to take?

A: The system considers:

- Threat severity (low, medium, high, critical)

- Threat source (ML detection, Snort, WebShield, Sandbox)

- Threat type (file, URL, network)

- User preferences and policies

- Automatic vs manual mode

Q: Can I change the action for a threat?

A: Yes. You can:

- Manually select a different action
- Override automatic decisions
- Bulk change actions for multiple threats
- Restore or allow previously blocked items

Q: What happens if an action fails?

A: If an action fails:

- The system logs the error
- Tries alternative methods
- Reports the failure to administrators
- Keeps the threat record for retry

Q: Are actions reversible?

A: Some actions are reversible:

- **Quarantine**: Can be restored
- **Restrict**: Permissions can be restored
- **Delete**: Cannot be reversed (unless in Recycle Bin)
- **Allow**: Can be changed to other actions

Q: Can I automate threat actions?

A: Yes. The system supports:

- Automatic actions based on threat severity
- Policy-based automation
- Custom action rules
- Scheduled actions

Summary

Contributor B's technical contributions form the core security infrastructure of AI Shield:

- **Quarantine System**: Safe isolation of suspicious files with recovery capability
- **Secure Delete**: Multiple deletion methods with cross-platform support
- **Permission Restriction**: OS-level file protection without moving files
- **Background Scanner**: Real-time file system monitoring and scanning
- **OS-Level Operations**: Low-level system integration for security tasks
- **Snort IDS Integration**: Network intrusion detection system integration
- **Network Monitoring**: Active connection tracking and IP blocking
- **URL Blocking**: Automatic and manual website blocking
- **Threat Action Logic**: Coordinated response system for threat management

All these systems work together to provide comprehensive security protection, from file-level threats to network-level attacks, with automated responses and manual override capabilities.

Technical Stack

- **Languages**: Python 3.8+

- **Libraries**:

- `watchdog` (file system monitoring)
- `send2trash` (safe deletion)
- `sqlite3` (database)
- `subprocess` (OS commands)

- **Platforms**: Windows, Linux, macOS

- **Integration**: FastAPI backend, SQLite database, WebSocket real-time updates

---